

Online Covering with Convex Objectives and Applications

Yossi Azar*
Tel-Aviv University

Ilan Reuven Cohen†
Tel-Aviv University

Debmalya Panigrahi‡
Duke University

Abstract

We give an algorithmic framework for minimizing general convex objectives (that are differentiable and monotone non-decreasing) over a set of covering constraints that arrive online. This substantially extends previous work on online covering for linear objectives (Alon *et al.*, STOC 2003) and online covering with offline packing constraints (Azar *et al.*, SODA 2013). To the best of our knowledge, this is the first result in online optimization for generic non-linear objectives; special cases of such objectives have previously been considered, particularly for energy minimization.

As a specific problem in this genre, we consider the unrelated machine scheduling problem with startup costs and arbitrary ℓ_p norms on machine loads (including the surprisingly non-trivial ℓ_1 norm representing total machine load). This problem was studied earlier for the makespan norm in both the offline (Khuller *et al.*, SODA 2010; Li and Khuller, SODA 2011) and online settings (Azar *et al.*, SODA 2013). We adapt the two-phase approach of obtaining a fractional solution and then rounding it online (used successfully to many linear objectives) to the non-linear objective. The fractional algorithm uses ideas from our general framework that we described above (but does not fit the framework exactly because of non-positive entries in the constraint matrix). The rounding algorithm uses ideas from offline rounding of LPs with non-linear objectives (Azar and Epstein, STOC 2005; Kumar *et al.*, FOCS 2005). Our competitive ratio is tight up to a logarithmic factor. Finally, for the important special case of total load (ℓ_1 norm), we give a different rounding algorithm that obtains a better competitive ratio than the generic rounding algorithm for ℓ_p norms. We show that this competitive ratio is asymptotically tight.

*Email: azar@tau.ac.il. Supported in part by the Israel Science Foundation (grant No. 1404/10) and by the Israeli Centers of Research Excellence (I-CORE) program, (Center No.4/11). Part of this work was done while the author was visiting Microsoft Research, Redmond.

†Email: ilanrcohen@gmail.com. Supported in part by the Israeli Centers of Research Excellence (I-CORE) program.

‡Email: debmalya@cs.duke.edu. Supported in part by a Duke University startup grant and a Google Faculty Research Award. Part of this work was done while the author was visiting Microsoft Research, Redmond.

1 Introduction

Positive linear programming (also known as packing/covering) with convex (non-linear) objectives model a wide range of problems in combinatorial optimization and operations research. In algorithmic theory, they have been used in many areas including machine scheduling [8], packet routing [3], energy minimization [14], etc. In this paper, we consider the problem of minimizing arbitrary convex functions under linear covering constraints that arrive online. This significantly generalizes and extends previous frameworks for online covering algorithms with linear objectives [2, 16] and with offline packing constraints [7]. For convex objectives that are monotone and differentiable, we give a simple deterministic online algorithm that guarantees a nearly optimal solution. Then, we consider a natural representative of this genre of problems in machine scheduling — minimize the ℓ_p norm of machine loads where each machine has a startup cost. This problem arises in the context of energy optimization in cloud computing, and was previously studied for the makespan norm of machine loads in both the offline [25, 28] and online [7] settings. We give an online algorithm for this problem based on a two-phase process (commonly used in the online setting for *linear* objectives) of obtaining a competitive fractional solution, and rounding it online. While our online framework for general convex objectives cannot be used directly,¹ we use the intuition that we gained from it to obtain an online fractional solution in the first phase. In the second phase, we combine ideas from offline rounding for ℓ_p objectives [8, 27] and online rounding for exponential objectives [7] in a novel manner to obtain an integral assignment of jobs to machines.

Online Covering with General objectives (OCG): The goal is to minimize a *convex, non-decreasing, differentiable* function $f(\mathbf{x})$ of m variables $\mathbf{x} = \langle x_1, x_2, \dots, x_m \rangle$ subject to n linear covering constraints $C\mathbf{x} \geq \mathbf{c}$ that arrive online. Here, C is an $n \times m$ matrix and \mathbf{c} is an n -dimensional vector, both with non-negative entries. The variables x_i , $1 \leq i \leq m$, are also constrained to be non-negative and must be monotonically non-decreasing over the course of the online algorithm. On the arrival of a new covering constraint, it must be satisfied by increasing the values of the variables (note that the monotonicity of the variables and non-negativity of the constraint matrix implies that all constraints previously satisfied continue to be satisfied). This framework generalizes the following settings:

- **Online Covering with Linear Objectives (OCL)** [16, 2]: This is the special case where the function $f(\mathbf{x})$ is a linear function. This problem, in turn, generalizes the fractional versions of several important problems such as online set cover [2], online non-metric facility location [1], online network design problems [1, 30, 22, 23], etc.
- **Online Mixed Packing and Covering (OMPC)** [7]: In this problem, there are two sets of constraints: a set of n linear covering constraints $C\mathbf{x} \geq \mathbf{c}$ that arrive online and a set of r linear packing constraints $P\mathbf{x} \leq \mathbf{p}$ that are given offline. All entries in C, P, \mathbf{c} , and \mathbf{p} are non-negative, and the variables x_i , $1 \leq i \leq m$, must be non-negative and monotonically non-decreasing over the course of the online algorithm. The goal here is to exactly satisfy all the covering constraints, and approximately satisfy all the packing constraints (the approximation is provably required). For convenience, let us define a new set of (derived) variables $\lambda = \langle \lambda_1, \lambda_2, \dots, \lambda_r \rangle$, where $\lambda_k = \frac{\sum_i P_{ki} x_i}{P_k}$. In other words, λ_k is the *violation*² for the k th packing constraint. Then, the objective is to minimize the maximum violation; i.e., $f(\mathbf{x}) = \max_k \lambda_k$. This objective, as stated, has a large ($O(r)$) measure of convexity (will be defined later) and hence it is not useful for the OCG framework. However, as shown in [7], the objective function can be modified to $f(\mathbf{x}) = \ln(\sum_k e^{\lambda_k})$ up to a loss of $O(\log r)$ in the competitive ratio. The new function satisfies the conditions of the OCG problem. More generally, we can also consider any ℓ_p norm of the vector λ as our function; this also generalizes [7] since the maximum violation is known to be within a constant factor of the $\ell_{\ln r}$ norm.

¹Some of the constraints are not packing/covering constraints, i.e., have negative coefficients.

²One may also define $\lambda_k = \max\left(\frac{\sum_i P_{ki} x_i}{P_k}, 1\right)$; our results hold also for this definition.

The second part of our paper focuses on a representative problem in the genre of online covering problems with non-linear objectives: **Unrelated Machine Scheduling with Startup Cost** (UMSC). Let M be a set of m machines, where machine i has *startup cost* $c_i \geq 0$, and J be a set of n jobs that arrive online. The *processing time* of job j on machine i is denoted $p_{ij} \geq 0$. A *schedule* is an assignment of jobs to machines, and the *load* L_i of a machine i in a schedule is the sum of processing times of all jobs assigned to it. The *open machines* M_o are the machines to which at least one job has been assigned and the cost of the schedule is the sum of startup costs of open machines. The goal is to obtain a schedule that simultaneously minimizes cost and some function f of machine loads. The typical functions for f are: (1) the *makespan* or maximum load among all machines, i.e., $f = \max_{i \in M} L_i$, (2) the total load over all machines, i.e., $f = \sum_{i \in M} L_i$, and (3) the more general ℓ_p -norm of the load, i.e., $f = (\sum_{i \in M} (L_i)^p)^{1/p}$ for any fixed $p \in [1, \log m]$ (since $\ell_p \equiv \ell_\infty$ for $p \geq \log m$). The existence of startup costs makes even the case of minimizing the total load (ℓ_1 norm) non-trivial since the machine on which a job runs the fastest might have a large cost. This forces the algorithm to strike a balance between opening machines that have large startup costs but can run jobs at high speeds and those that have smaller startup costs but run slower.

Note: We assume that the UMSC input includes a pair of values (\mathbf{C}, \mathbf{L}) with the guarantee that there exists a schedule of cost at most \mathbf{C} and ℓ_p -norm at most \mathbf{L} (p is fixed). Using standard doubling guesses, our formulation can be shown to be asymptotically equivalent to one where one objective needs to be optimized subject to a given bound on the other. Moreover, our formulation subsumes single objective formulations where the two objectives are combined using a linear function.

The UMSC problem is closely connected to energy management in data centers, which has recently emerged as one of the most important practical challenges in cloud computing (see, e.g., [15] for a discussion). With this motivation, the problem was studied in the offline setting for the makespan norm [25, 19, 28] and in the online setting [7]. In this paper, we extend this line of work significantly to all ℓ_p norms, including the surprisingly non-trivial ℓ_1 norm representing total machine loads. Note that the UMSC problem generalizes the online set cover problem [2] (for $p_{ij} = 0$ or ∞) and the online unrelated machine scheduling problem [4, 5] (for $c_i = 0$). A similar energy minimization problem (call it *online covering for energy minimization* or OCE) was studied in [21] which can be thought of as the UMSC problem with assignment costs instead of startup costs. This seemingly minor difference, however, completely changes the structure of the problem since the OCE problem does *not* generalize set cover. In fact, perhaps the most illustrative difference between the two problems is for the case of linear loads, where UMSC remains non-trivial whereas a greedy algorithm suffices for OCE. Moreover, the goal in [21] was to only obtain a fractional solution whereas we are interested in an integral solution and therefore need to consider integrality gaps.

1.1 Our Results

The OCG framework. We will denote the maximum and minimum non-zero entry in the constraint matrix C by c_{\max} and c_{\min} respectively. Our result also depends on two parameters of the objective function f . The

first parameter $\beta = \max_{\mathbf{x}} \frac{\sum_{i=1}^m x_i \cdot \frac{\partial f}{\partial x_i}}{f(\mathbf{x})}$. Informally, this is a measure of the convexity of the function: e.g.,

$\beta = O(1)$ for any polynomial function but infinite for exponential functions. The second parameter γ is the smallest positive number such that $f(1/\gamma, \dots, 1/\gamma) \leq \text{OPT}$. To understand the dependence on γ , consider an objective $f(x_1, x_2) = 0$ if $x_1 = 0$ or $x_2 = 0$ but > 0 otherwise. For this objective, it is impossible to obtain a finite competitive ratio¹ and this is encapsulated by an infinite value of γ .

We are now ready to state our result.

Theorem 1. *There is a deterministic online algorithm for the OCG problem that produces a fractional solution with objective at most $f(\beta \log(\gamma/c_{\min}) \mathbf{x}^*) + \beta f(\mathbf{x}^*)$, where \mathbf{x}^* is any optimal solution. In particular,*

¹To see this, let the first constraint be $x_1 + x_2 \geq 1$. If the online algorithm sets $x_1 > 0$ (resp., $x_2 > 0$) in response to this constraint, then the next constraint is $x_2 \geq 1$ (resp., $x_1 \geq 1$).

for sub-homogeneous functions (i.e., functions satisfying $f(\eta \mathbf{x}) \leq \eta f(\mathbf{x})$ for any $\eta > 1$) the competitive ratio is $O(\beta \log(\gamma/c_{\min}))$.

First, we apply Theorem 1 to a linear objective $\sum_{i=1} a_i x_i$, i.e., the OCL problem [16]. We note that any variable x_i for which $a_i/c_{\max} > \text{OPT}$ can be discarded at the outset. After discarding these variables, we can set $\gamma = mc_{\max}$, and the competitive ratio is $O(\log(mc_{\max}/c_{\min}))$ since $\beta = 1$. For $\{0, 1\}$ constraint matrices (e.g. the fractional set cover problem [2] and network design problems in [1, 30, 22, 23]), the competitive ratio is $O(\log m)$.

Next, we consider the OMPC problem with the ℓ_p norm objective, i.e., $f(\mathbf{x}) = (\sum_k (\lambda_k)^p)^{1/p}$ (recall that λ_k denotes the violation of the k th packing constraint). Let p_{\max} and p_{\min} be the maximum and minimum non-zero entries in the packing matrix P respectively, and let $\kappa = p_{\max}/p_{\min}$; similarly, let $\rho = c_{\max}/c_{\min}$. Also, let $d \leq m$ denote the maximum number of variables in any packing or covering constraint. In order to apply Theorem 1, we set $\gamma = d \cdot c_{\max} \cdot (p_{\max}/p_{\min})$, since for any packing constraint k , we have $\sum_{i=1}^m p_{ki} x_i^0 \leq p_{\min}/c_{\max} \leq \sum_{i=1}^m p_{ki} x_i^*$. Also, $\beta = p$ for the ℓ_p norm function, yielding the following corollary (note that, it is enough to consider $p \leq \log r$ since $\ell_p \approx \ell_{\log r}$ for any $p \geq \log r$, including the ℓ_∞ norm).

Corollary 2. *There is a deterministic online algorithm for the OMPC problem with ℓ_p norm that has a competitive ratio of $O(p \log(d\rho\kappa))$. For $\{0, 1\}$ constraint matrices, the competitive ratio is $O(p \log d)$.*

This matches the upper bound of $O(\log r \cdot \log(d\rho\kappa))$ for the ℓ_∞ norm ($\max_k \lambda_k$) in [7] by using $p = \log r$ since the $\ell_{\log r}$ norm approximates the ℓ_∞ norm up to a small constant. Alternatively, one may try to apply Theorem 1 directly for the function $f(\mathbf{x}) = \max_k \lambda_k$. However, this results in a worse approximation ratio since for this function, $\beta = r$. In fact, the authors in [7] used a third function $f(\mathbf{x}) = \ln(\sum_k e^{\lambda_k})$ as the surrogate objective for $\max_k \lambda_k$. Theorem 1 can be directly applied to this function as well, yielding a matching result to those obtained by the $\ell_{\log r}$ norm in Corollary 2 and in [7].

We also show that Corollary 2 is nearly tight, by adapting a lower bound in [7] to the ℓ_p norm.

Theorem 3. *Any deterministic algorithm for OMPC with respect to the ℓ_p norm on λ is $\Omega(p \log(d/\log r))$ -competitive for $p \leq \log r$, even for $\{0, 1\}$ constraint matrices.*

The UMSC problem. Following standard convention, we say that a randomized algorithm for the UMSC problem has a bi-criteria competitive ratio of (α, β) if it produces a schedule of expected cost at most αC and the expected ℓ_p norm of the load is at most βL . Our main result is a randomized algorithm that proves the next theorem.

Theorem 4. *There is a randomized online algorithm for the UMSC problem for arbitrary fixed p with a competitive ratio of $(O(\log m \log(mn)), O(p^2 \log^{1/p}(mn)))$.*

Since $p \leq \log m$, our competitive ratio is upper bounded by $(O(\log m \log(mn)), O(\log^2 m \log^{1/p}(mn)))$.

Recall that the UMSC problem generalizes the set cover problem [2] and the unrelated machine scheduling problem for ℓ_p norms [5, 17]. The lower bound for the UMSC problem is derived from lower bounds for these problems (see [2, 26] for the cost lower bound derived from online set cover and [5, 17] for the ℓ_p -norm lower bound derived from online unrelated machine scheduling).

Observation 5. *No algorithm for the UMSC problem can have a competitive ratio of $o(p)$ in the ℓ_p -norm of machine loads. Further, under standard complexity assumptions, no algorithm for this problem can have a competitive ratio of $o(\log m \log n)$ in the cost of the schedule.*

It follows from these lower bounds that the competitive ratios in Theorem 4 are almost tight in both objectives.

We also separately consider the important special case of $p = 1$, where the goal is to minimize the sum of all machines loads. For this case, Theorem 4 gives a competitive ratio of $(O(\log m \log(mn)), O(\log(mn)))$. We improve this result and obtain a tight (up to constants) competitive ratio in both objectives.

Theorem 6. *There is a randomized online algorithm for the UMSC problem for $p = 1$ with a competitive ratio of $(O(\log m \log n), O(1))$.*

1.2 Our Techniques

To solve the OCG problem, we use a continuous algorithm where the values smoothly increase over time. (The algorithm can be discretized for polynomial implementation, but the continuous version is easier to describe.) The rate of increase of each variable is inversely proportional the current partial derivative of the objective for this variable. Note that this extends the algorithm for online set cover [2] where the partial derivative is the cost of the set. In the analysis, we implicitly use the Lagrangian dual of the convex objective. The algorithm increases the dual variable of the current constraint at unit rate (as in [2, 16]). The analysis establishes approximate stationarity of the optimal solution, and a relationship between the growth of the primal objective and the Lagrangian dual. These two facts are coupled to bound the value of the objective in the algorithmic solution by that of any suitably scaled feasible solution, thereby showing Theorem 1.

For the UMSC problem, using the syntactic definition of the ℓ_p -norm (we actually use the ℓ_p^p norm for ease of manipulation) as the objective function leads to a polynomial integrality gap. Consider the following simple example. Suppose there are m machines with startup cost 1 each and m jobs arrive with $p_{ij} = 1$ for each (i, j) -pair. Also, let $\mathbf{C} = 1$. Then, a feasible fractional solution is to open each machine to $x_i = 1/m$ and set $y_{ij} = 1/m$ for each (i, j) -pair. While the objective value of this fractional solution is m , any integer solution with a poly-logarithmic competitive ratio in the cost (recall that this is what we are aiming for) can open at most a poly-logarithmic number of machines, and therefore will have an objective value of at least $\log m(m/\log m)^p$. To overcome this integrality gap, we refine our definition of the ℓ_p -norm of the load on a partially open machine (for a fully open machine, we continue to use the syntactic definition of $\sum_{i \in M} (\sum_{j \in J} p_{ij} y_{ij})^p$) to $\sum_{i \in M} \left(\frac{\sum_{j \in J} p_{ij} y_{ij}}{x_i} \right)^p x_i$, where y_{ij} is the assignment of job j to machine i and x_i is the fraction to which machine i is open. (We use constraints $y_{ij} \leq x_i$ which deviates from positive LPs as stated above.)

However, there is still a large integrality gap since a fractional solution can split a large job into several small jobs and distribute them on multiple machines. In order to overcome it, we add an extra term $\sum_{i \in M} (\sum_{j \in J} y_{ij} p_{ij}^p)$ to the objective function (see also [27, 8]). Note that for an integer solution, this additional term is bounded above by the actual ℓ_p^p norm. The complete LP is given in Fig. 1.

$$\begin{aligned}
 \text{Minimize} \quad & \sum_{i \in M} \left(\frac{\sum_{j \in J} p_{ij} y_{ij}}{x_i} \right)^p x_i + \sum_{i \in M} \left(\sum_{j \in J} y_{ij} p_{ij}^p \right) \quad \text{subject to} \\
 & \sum_{i \in M} c_i x_i \leq \mathbf{C} \tag{1} \\
 & y_{ij} \leq x_i \quad \forall i \in M, j \in J \tag{2} \\
 & \sum_{i \in M} y_{ij} \geq 1 \quad \forall j \in J \tag{3} \\
 & x_i, y_{ij} \in [0, 1] \tag{4}
 \end{aligned}$$

Figure 1: The UMSC LP

To obtain a fractional solution for this formulation, we design a non-linear potential function that guides multiplicative updates of the variables. For partially open machines, the potential function is defined according to the fractional cost of the machine; during this phase, the primary goal of the algorithm is cost minimization. The multiplicative update steps are designed such that the load on the machine is “small” in this phase. Once x_i increases to 1, i.e., machine i is *fully open*, the potential function is defined on the ℓ_p -norm of the fractional load on the machine. In this phase, the primary goal of the algorithm shifts to load minimization. In bounding the ℓ_p -norm of the load, we also use ideas due to Caragiannis [17], who gave an elegant analysis for the problem without startup costs.

1.3 Previous Work

Packing and covering have been widely used and analyzed in offline scenarios, typically for linear objectives (e.g. [31, 20]). In a sequence of recent papers, online versions of these problems have also been studied including online set cover [1], network design [2, 30, 22, 23], paging [12, 13, 11, 10], general online covering *or* online packing constraints [16], online covering constraints *and* offline packing constraints [7], etc. Non-linear objectives have also been considered for specific problems, especially related to energy minimization (e.g., [21]). To the best of our knowledge, this is the first paper to give results for optimizing general non-linear objectives under linear constraints.

Assigning jobs that arrive online to unrelated machines so as to minimize the ℓ_p -norm of machine loads is a central question in scheduling theory. For $p = 1$, the natural greedy strategy of assigning each job to the machine on which it runs the fastest is optimal, but for $p > 1$, the problem turns out to be more challenging. For the makespan objective (maximum load or the ℓ_∞ norm, which is also asymptotically equivalent to any ℓ_p norm with $p \geq \log m$), Aspnes *et al.* [4] obtained a competitive ratio of $O(\log m)$, which is asymptotically tight [9]. For any $p \leq \log m$, Awerbuch *et al* [5] obtained a tight competitive ratio of $O(p)$. Subsequently, Caragiannis [17, 18] provided an elementary analysis for this algorithm, while also tightening the constants in the upper and lower bounds. Various other models and objectives have been considered for the load balancing problem; the interested reader is referred to surveys such as [6, 33, 32, 34].

The offline version of UMSC with the makespan objective was introduced by Khuller *et al.* [25], where they gave an $O(2(1 + 1/\varepsilon)(1 + \ln(n/OPT)), 2 + \varepsilon)$ -approximation algorithm for any $\varepsilon > 0$. (For further work on this problem, see [19, 28]). The online version of this problem with the makespan objective was considered in [7], who obtained a poly-logarithmic bicriteria competitive ratio. We significantly generalize these results by considering ℓ_p -norms for arbitrary values of p .

Roadmap. The algorithm for OCG (Theorem 1) is in Section 2. The fractional algorithm and the randomized rounding procedure for UMSC with general ℓ_p norms (Theorem 4) are in Sections 3 and 4 respectively. The lower bound for OMPC (Theorem 3) is given in the appendix.

2 Algorithm for the OCG problem

We consider the convex program for an m -dimensional *non-negative* variable $\mathbf{x} = \langle x_i : 1 \leq i \leq m \rangle$:

$$\text{minimize } f(\mathbf{x}) \text{ subject to } C\mathbf{x} \geq \mathbf{1},$$

where the objective function f is *convex*, *monotone non-decreasing*, and *differentiable everywhere*. The covering matrix C is an $m \times n$ -dimensional non-negative matrix (the (i, j) th entry is denoted c_{ij}) and the RHS is wlog (by scaling) the all-ones vector in n dimensions. The constraints arrive online and must be satisfied when they arrive. The variable \mathbf{x} has to be monotone non-decreasing over time in every dimension. It will also be convenient to define the Lagrangian dual:

$$L(\mathbf{x}, \mathbf{y}) = f(\mathbf{x}) - \mathbf{y} \cdot (C\mathbf{x} - \mathbf{1}).$$

2.1 Description of the Algorithm

We define a continuous algorithm where \mathbf{x} is initialized to a certain value and smoothly increases over time. For a polynomial implementation, this algorithm can be discretized by choosing a small enough discrete “step size”.

We initialize \mathbf{x} to the vector $\mathbf{x}^0 = (1/\gamma, 1/\gamma, \dots, 1/\gamma)$, where γ is large enough so that

$$f(1/\gamma, 1/\gamma, \dots, 1/\gamma) \leq \text{OPT}.$$

When a constraint $\sum_i c_{ij}x_i \geq 1$ arrives online, we increase \mathbf{x} at the following rate until the constraint is satisfied:

$$\forall i \in [m], \frac{dx_i}{dt} = \frac{c_{ij}x_i}{\left(\frac{\partial f}{\partial x_i}\right)}.$$

For the analysis, we also increase the dual variable y_j at the rate $\frac{dy_j}{dt} = 1$.

2.2 Analysis of the Algorithm

The first observation follows from our choice of γ .

Observation 7. *The value of the objective $f(\mathbf{x})$ after the initialization is at most OPT.*

Our main goal is to bound the total increase of the objective over the course of the online algorithm. Recall the KKT conditions for optimality of convex programs:

1. **Feasibility:** $C\mathbf{x} \geq \mathbf{1}$, $\mathbf{x} \geq \mathbf{0}$, and $\mathbf{y} \geq \mathbf{0}$;
2. **Complementary Slackness:** $y_j \cdot (\sum_{i=1}^m c_{ij}x_i - 1) = 0$ for all $j \in [n]$;
3. **Stationarity:** $\sum_{j=1}^n c_{ij}y_j = \frac{\partial f}{\partial x_i}$ for all $i \in [m]$.

Clearly, the online algorithm maintains feasibility (condition 1). It will be useful to establish approximate stationarity (condition 3) at the end of the algorithm.

Lemma 8. *Let $\alpha = \ln(\gamma/c_{\min})$ where $c_{\min} = \min_{i,j}\{c_{ij} > 0\}$, and let $\left(\frac{\partial f}{\partial x_i}\right)_e$ be the value of $\left(\frac{\partial f}{\partial x_i}\right)$ at the end of the algorithm. The following holds for all $i \in [m]$:*

$$\sum_{j=1}^n c_{ij}y_j \leq \alpha \cdot \left(\frac{\partial f}{\partial x_i}\right)_e. \quad (5)$$

Proof. Suppose the algorithm is updating variables for constraint j at time t . We bound the rate of increase of the LHS of (5):

$$\frac{d\sum_{j=1}^n c_{ij}y_j}{dt} = \frac{dc_{ij}y_j}{dt} = c_{ij} = \left(\frac{\partial f}{\partial x_i}\right)_t \cdot (1/x_i) \cdot \frac{dx_i}{dt} \leq \left(\frac{\partial f}{\partial x_i}\right)_e \cdot (1/x_i) \cdot \frac{dx_i}{dt}.$$

The last step uses the convexity of f , which implies non-decreasing partial derivatives. Since the maximum value of any variable x_i can be $1/c_{\min}$, it follows that

$$\sum_{j=1}^n c_{ij}y_j \leq \left(\frac{\partial f}{\partial x_i}\right)_e \int_{1/\gamma}^{1/c_{\min}} \frac{dx_i}{x_i}. \quad \square$$

We start the analysis by comparing the Lagrangian dual to the primal objective.

Lemma 9. *At any stage of the online algorithm,*

$$f(\mathbf{x}) - f(\mathbf{x}^0) \leq \sum_{j=1}^n y_j. \quad (6)$$

Proof. We compare the rates of increase of the two sides of Eqn. 6 in the online algorithm:

$$\frac{df(\mathbf{x})}{dt} = \sum_{i=1}^m \left(\frac{\partial f}{\partial x_i}\right)_t \cdot \frac{dx_i}{dt} = \sum_{i=1}^m \left(\frac{\partial f}{\partial x_i}\right)_t \cdot \frac{c_{ij}x_i}{\left(\frac{\partial f}{\partial x_i}\right)_t} = \sum_{i=1}^m c_{ij}x_i \leq 1 = \frac{dy_j}{dt}. \quad \square$$

We are now ready to prove our main lemma.

Lemma 10. *If \mathbf{x}^* be any feasible solution and \mathbf{x} is the solution obtained by the online algorithm, then*

$$f(\mathbf{x}) \leq f(\alpha\beta\mathbf{x}^*) + \beta f(\mathbf{x}^0), \quad \text{where } \beta = \max_{\mathbf{x}} \frac{\sum_{i=1}^m x_i \cdot \frac{\partial f}{\partial x_i}}{f(\mathbf{x})}.$$

Proof. By first order convexity properties,

$$f(\alpha\beta\mathbf{x}^*) - f(\mathbf{x}) \geq \sum_{i=1}^m (\alpha\beta x_i^* - x_i) \frac{\partial f}{\partial x_i}.$$

The RHS above can be written as

$$\beta \sum_{i=1}^m \left(\alpha x_i^* \frac{\partial f}{\partial x_i} - (x_i/\beta) \frac{\partial f}{\partial x_i} \right) \geq \beta \sum_{i=1}^m \left(x_i^* \sum_{j=1}^n c_{ij} y_j - (x_i/\beta) \frac{\partial f}{\partial x_i} \right) \quad (\text{by Lemma 8}).$$

Swapping summations, the RHS above can be written as

$$\beta \left(\sum_{j=1}^n y_j \sum_{i=1}^m c_{ij} x_i^* - (1/\beta) \sum_{i=1}^m x_i \frac{\partial f}{\partial x_i} \right) \geq \beta \left(\sum_{j=1}^n y_j - (1/\beta) \sum_{i=1}^m x_i \frac{\partial f}{\partial x_i} \right) \quad (\text{by feasibility of } \mathbf{x}^*).$$

Using the definition of β , the RHS above can be written as

$$\beta \left(\sum_{j=1}^n y_j - f(\mathbf{x}) \right) \geq -\beta f(\mathbf{x}^0) \quad (\text{by Lemma 9}). \quad \square$$

Finally, Theorem 1 follows from Lemma 10 and Observation 7.

3 Fractional Algorithm for UMSC

Recall that the input contains the pair of values (\mathbf{C}, \mathbf{L}) with the guarantee that there exists a feasible assignment of cost at most \mathbf{C} and ℓ_p -norm at most \mathbf{L} . We will fix such an assignment and call it the *optimal* solution (denoted OPT). We will also assume that the algorithm knows the number of jobs n , which is without loss of generality up to constant factors in the competitive ratio.

The algorithm has two phases — an offline pre-processing phase, and an online phase that (fractionally) schedules the arriving jobs.

Offline Pre-processing. First, we note that all machines whose startup cost exceeds \mathbf{C} are unused in OPT; hence, the algorithm discards these machines at the outset. Let m be redefined to the number of machines with startup cost at most \mathbf{C} . Next, we multiply the costs of all machines by $\frac{m}{\mathbf{C}}$ so that the cost of OPT is m . For any machine i with $c_i \leq 1$, we set $c_i = 1$; this increases the optimal cost to at most $2m$. We initialize x_i as follows: if $c_i = 1$, we set $x_i = 1$; else ($1 < c_i \leq m$), we set $x_i = 1/m$. Finally, we multiply all processing times by $\frac{\beta^{1/p}}{\mathbf{L}}$, where $\beta = \frac{m \ln m}{(40p)^p}$; then an ℓ_p^p -norm of β with the scaled processing times implies an ℓ_p -norm of \mathbf{L} with the original processing times.

Before describing the online phase, we need to introduce some notation. Let machine i be said to be *closed*, *partially open*, or *fully open* depending on whether $x_i = 0$, $0 < x_i < 1$ or $x_i = 1$ respectively. We distinguish between (fractions of) jobs that are assigned when a machine is partially open and those that are assigned when the machine is fully open; let us denote the respective sets of jobs $J_0^{(i)}$ and $J_1^{(i)}$. (There can be at most one job that is in both sets since machine i became fully open while the job was being assigned.

For this job, we will consider the fraction of the job assigned while machine i was partially open as being in set $J_0^{(i)}$ and the remainder in set $J_1^{(i)}$. Recall that the load on machine i is $L_i = \sum_{j \in J} y_{ij} p_{ij}$. However, for partially open machines, calculating this load exactly turns out to be difficult. Instead, we maintain an upper bound of $c_i^{1/p} x_i$ on the load, which then allows us to define a proxy load $\tilde{L}_i = c_i^{1/p} x_i + \sum_{j \in J_1^{(i)}} y_{ij} p_{ij}$.

Suppose the algorithm wants to assign an infinitesimal fraction of a job to the machines. Intuitively, it should prefer machines whose cost and fractional ℓ_p -norm increases the least on assigning the fractional job. To formalize this notion, we define a function ψ_{ij} that the algorithm uses to sort machines in increasing order of preference when assigning a fraction of job j :

$$\psi_{ij} = \begin{cases} \max\{c_i^{(p-1)/p} p_{ij}, p_{ij}^p\} & \text{if } x_i < 1. \\ (\tilde{L}_i + p_{ij})^p - \tilde{L}_i^p & \text{if } x_i \geq 1. \end{cases}$$

Online Assignment. When a new job j arrives, we use Algorithm 1 to update x_i, y_{ij} in multiple steps until $\sum_{i \in M} y_{ij} = 1$. This is a polynomial-time implementation of a continuous multiplicative weight augmentation algorithm, N being the discretization parameter that we set to $nm \ln m$ to ensure that each discrete step is small enough. (For technical reasons, we maintain $y_{ij} \leq 2x_i$ instead of $y_{ij} \leq x_i$.)

while $\sum_{i \in M} y_{ij} < 1$, do the following:

- Sort the machines in non-increasing order by ψ_{ij} and let $P(j)$ be the minimal prefix¹ of this sorted order such that $\sum_{i \in P(j)} x_i \geq 1$.
- For each partially² open machine $i \in P(j)$, set $\Delta x_i = \frac{x_i}{c_i N}$.
- For each machine $i \in P(j)$, set $\Delta y_{ij} = \min\left(\frac{x_i}{\psi_{ij} N}, 2x_i - y_{ij}\right)$.
- Update $x_i \leftarrow x_i + \Delta x_i$, $y_{ij} \leftarrow y_{ij} + \Delta y_{ij}$, unless x_i or y_{ij} exceeds 1. In this case, we do a *small step*, i.e., we redefine Δx_i and Δy_{ij} with a value of $N' > N$ instead of N so that $\max_{i,j} \{x_i, y_{ij}\} = 1$.

Algorithm 1: Fractional assignment for a single job

3.1 Analysis of the fractional algorithm

We bound the cost and ℓ_p -norm of the fractional algorithm using a potential function defined as

$$\Phi_i = \begin{cases} c_i x_i & \text{if } x_i < 1. \\ \tilde{L}_i^p + \sum_{j \in J_1^{(i)}} y_{ij} p_{ij}^p & \text{if } x_i = 1. \end{cases}$$

The overall potential function $\Phi = \sum_{i \in M} \Phi_i$. Note that the potential function is continuous and monotonically non-decreasing. First, observe that the potential of a partially open machine is exactly its fractional startup cost and becomes c_i when the machine is fully opened (i.e., when x_i becomes 1). Therefore, by monotonicity, $\Phi_i \geq c_i x_i$ during the entire run of the algorithm. Additionally, the algorithm ensures for each partially open machine, the following conditions are satisfied:

$$\sum_{j \in J_0^{(i)}} y_{ij} p_{ij} \leq c_i^{1/p} x_i \quad \text{and} \quad \sum_{j \in J_0^{(i)}} y_{ij} p_{ij}^p \leq c_i x_i. \quad (7)$$

¹ $P(j)$ is always defined since $\sum_{i \in M} x_i \geq 1$.

²Only the last machine in $P(j)$ may be fully open; all other machines are partially open.

Therefore, the potential also bounds the fractional objective function, i.e. the fractional ℓ_p^p -norm of the load.

Note that ψ_{ij} is a bound on the discrete differential $\frac{\Delta\Phi_i}{\Delta y_{ij}}$. For partially open machines, $\frac{\Delta\Phi_i}{\Delta y_{ij}} = \frac{c_i \Delta x_i}{\Delta y_{ij}}$, and from the two conditions in Eqn. 7 we get $\Delta y_{ij} p_{ij} \leq c_i^{1/p} \Delta x_i$, and $\Delta y_{ij} p_{ij}^p \leq c_i \Delta x_i$, which defines ψ_{ij} . For fully open machines, the discrete differential is immediate.

First, we bound the increase in potential in the pre-processing phase (Lemma 11), in each single step (Lemma 12), and in all the *small steps* (Lemma 13).

Lemma 11. *At the end of the pre-processing phase, $\Phi \leq m$.*

Proof. After pre-processing, the potential $\Phi = \sum_{i \in M} c_i x_i$, where each $c_i x_i \leq 1$. □

Lemma 12. *The increase in the potential in a single algorithmic step is at most $5/N$.*

Proof. The total increase in Φ for partially open machines in each step is

$$\sum_{i \in PA} c_i \Delta x_i \leq \sum_{i \in P(j)} c_i \frac{x_i}{c_i N} \leq \frac{2}{N}.$$

For a fully open machine i , $\Delta y_{ij} = \frac{1}{N((\tilde{L}_i + p_{ij})^p - \tilde{L}_i^p)} \leq \frac{1}{pNp_{ij}\tilde{L}_i^{p-1}}$, which increases the first term in Φ by

$$\begin{aligned} (\tilde{L}_i + \Delta y_{ij} p_{ij})^p - \tilde{L}_i^p &\leq \left(\tilde{L}_i + \frac{1}{Np\tilde{L}_i^{p-1}} \right)^p - \tilde{L}_i^p \\ &= \tilde{L}_i^p \left(\left(1 + \frac{1}{Np\tilde{L}_i^p} \right)^p - 1 \right) \leq \tilde{L}_i^p \left(\left(1 + \frac{2}{N\tilde{L}_i^p} \right) - 1 \right) \leq \frac{2}{N}. \end{aligned}$$

The penultimate inequality follows from $(1 + \alpha)^p \leq 1 + 2\alpha p$ for $\alpha \leq 1/(2p)$, which in turn holds since $\tilde{L}_i^p \geq c_i \geq 1$ and $N \geq 2$. Additionally, note that

$$\Delta y_{ij} = \frac{1}{N((\tilde{L}_i + p_{ij})^p - \tilde{L}_i^p)} \leq \frac{1}{Np_{ij}^p}.$$

So the increase in the second term of Φ_i is at most $1/N$. Further, in each step, the load on at most one fully open machine increases. Hence, the total increase in potential is at most $5/N$. □

Lemma 13. *The total increase in potential in all the small steps is at most 2.*

Proof. In each small step, either machine becomes fully open or a job is completely assigned. So, the total number of small steps is at most $n + m$. Therefore, by Lemma 12, the total increase in potential in the small steps is at most $\frac{m+n}{N} < \frac{n+m}{nm} \leq 2$. □

This leaves us with the task of bounding the total number of regular (i.e., not small) steps. We classify these steps according to an optimal solution (denoted OPT). Let M_{OPT} denote the set of open machines in OPT and $\text{OPT}(j) \in M_{\text{OPT}}$ be the machine where job j is assigned to by OPT. The three categories are:

1. $\text{OPT}(j) \in P(j)$ and $\text{OPT}(j)$ is partially open
2. $\text{OPT}(j) \notin P(j)$ and $\text{OPT}(j)$ is partially open
3. $\text{OPT}(j)$ is fully open

We bound the total increase in potential in each of the three categories separately.

Lemma 14. *The total increase in potential in the first category steps is $O(m \log m)$.*

Proof. In any step of the first category, the value of $x_{\text{OPT}(j)}$ increases to $x_{\text{OPT}(j)} \left(1 + \frac{1}{c_{\text{OPT}(j)}N}\right)$. Since x_i is initialized to at least $1/m$ for every machine i in the pre-processing phase and x_i cannot exceed 1, it follows that the total number of steps in the first category is at most

$$\sum_{i \in M_{\text{OPT}}} c_i N \log m = O(Nm \log m).$$

Using Lemma 12, we conclude that the increase in potential in these steps is $O(m \log m)$. \square

Lemma 15. *The total increase in potential in the second category steps is $O(m \log m)$.*

Proof. For any step, let $Q(j)$ denote the set of machines in $P(j)$ for which $\Delta y_{ij} = 2x_i - y_{ij}$, and let $R(j) = P(j) \setminus Q(j)$. Note that for any job j , an algorithmic step for which $\sum_{i \in Q(j)} x_i \geq 1/2$ must be its last step. This follows from the observation that in this step, $\sum_{i \in M} (y_{ij} + \Delta y_{ij}) \geq \sum_{i \in Q(j)} (y_{ij} + \Delta y_{ij}) = \sum_{i \in Q(j)} 2x_i \geq 1$. So, there are at most n steps of this kind.

Now, we bound the number of algorithmic steps where $\sum_{i \in R(j)} x_i \geq 1/2$. In any such step, using the fact that $\psi_{ij} \leq \psi_{\text{OPT}(j)j}$ (otherwise $\text{OPT}(j) \in P(j)$), we have

$$\sum_{i \in M} \Delta y_{ij} \geq \sum_{i \in R(j)} \Delta y_{ij} = \sum_{i \in R(j)} \frac{x_i}{N \psi_{ij}} \geq \frac{1}{2N \psi_{\text{OPT}(j)j}}.$$

Since $\text{OPT}(j)$ is partially open, $\psi_{\text{OPT}(j)j} = \max\{c_{\text{OPT}(j)}^{(p-1)/p} p_{\text{OPT}(j)j}, p_{\text{OPT}(j)j}^p\}$. Let L_i^{OPT} be the load on machine i in OPT. Summing over all jobs, we have

$$\begin{aligned} \sum_{j \in J} \psi_{\text{OPT}(j)j} &\leq \sum_{j \in J} \left(c_{\text{OPT}(j)}^{(p-1)/p} p_{\text{OPT}(j)j} + p_{\text{OPT}(j)j}^p \right) \leq \sum_{j \in J} c_{\text{OPT}(j)}^{(p-1)/p} p_{\text{OPT}(j)j} + \beta \\ &= \sum_{i \in M} \sum_{j: \text{OPT}(j)=i} c_i^{(p-1)/p} p_{ij} + \beta = \sum_{i \in M} c_i^{(p-1)/p} L_i^{\text{OPT}} + \beta \leq m^{(p-1)/p} \beta^{\frac{1}{p}} + \beta \\ &\leq m^{(p-1)/p} \left(\frac{m \log m}{(40p)^p} \right)^{1/p} + \beta \leq \frac{m \log^{1/p} m}{40p} + \beta \leq 2m \log m, \end{aligned}$$

where we use Hölder's inequality (see e.g., [35]) in the first inequality on the second line. Therefore, the total number of steps in this category is bounded by $O(Nm \log m)$. By Lemma 12, the total increase in potential in these steps is $O(m \log m)$. \square

Lemma 16. *The total increase in potential in the third category steps is $O(m \log m)$.*

Proof. Define L_i^* as \tilde{L}_i at the end of the run of the algorithm. For each fully open machine i define $\psi_{ij}^* = (L_i^* + p_{ij})^p - L_i^{*p}$. By convexity of x^p , we have $\psi_{ij} \leq \psi_{ij}^*$. Recall the proof of Lemma 15 and the definition $R(j)$. An identical argument shows that for each step in the third category we have,

$$\sum_{i \in M} \Delta y_{ij} \geq \sum_{i \in R(j)} \Delta y_{ij} = \sum_{i \in R(j)} \frac{x_i}{N \psi_{ij}} \geq \frac{1}{2N \psi_{\text{OPT}(j)j}} \geq \frac{1}{2N \psi_{\text{OPT}(j)j}^*}.$$

Therefore, by summing over all jobs, the total number of the third category steps is $\sum_{j \in J} 2N \psi_{\text{OPT}(j)j}^*$. By Lemma 12, the total increase in potential in third category steps is $10 \left(\sum_{j \in J} \psi_{\text{OPT}(j)j}^* \right)$. Define $\Delta_o \Phi$ as the increase in potential first and second category steps along with the small steps, and Φ_0 to be the potential

after pre-processing. Also, let $\Delta_3\Phi$ be the increase in potential in third category steps and M_o be the set of machines that are fully opened by the algorithm. Then,

$$\begin{aligned}\Delta_3\Phi &\leq 10 \left(\sum_{j \in J} \psi_{\text{OPT}(j)j}^* \right) \leq \left(10 \sum_{j \in J} \left((L_{\text{OPT}(j)}^* + p_{\text{OPT}(j)j})^p - L_{\text{OPT}(j)}^{*p} \right) \right) \\ &\leq 10 \left(\sum_{i \in M_{\text{OPT}} \cap M_o} \sum_{j: \text{OPT}(j)=i} ((L_i^* + p_{ij})^p - L_i^{*p}) \right) \leq 10 \left(\sum_{i \in M_{\text{OPT}} \cap M_o} ((L_i^* + L_i^{\text{OPT}})^p - L_i^{*p}) \right).\end{aligned}$$

Rearranging the terms,

$$\begin{aligned}\left(\frac{\Delta_3\Phi}{10} + \sum_{i \in M_{\text{OPT}} \cap M_o} (L_i^*)^p \right)^{1/p} &\leq \left(\sum_{i \in M_{\text{OPT}} \cap M_o} (L_i^* + L_i^{\text{OPT}})^p \right)^{1/p} \leq \left(\sum_{i \in M_{\text{OPT}} \cap M_o} (L_i^* + L_i^{\text{OPT}})^p \right)^{1/p} \\ &\leq \left(\sum_{i \in M_{\text{OPT}} \cap M_o} L_i^{*p} \right)^{1/p} + \left(\sum_{i \in M_{\text{OPT}} \cap M_o} (L_i^{\text{OPT}})^p \right)^{1/p} \leq \left(\sum_{i \in M_{\text{OPT}} \cap M_o} L_i^{*p} \right)^{1/p} + \beta^{1/p}.\end{aligned}$$

Now, we have two cases. First, suppose $2(\Delta_3\Phi) > \sum_{i \in M_{\text{OPT}} \cap M_o} L_i^{*p}$. Then, we have

$$\begin{aligned}\left(\frac{\Delta_3\Phi}{10} + \sum_{i \in M_{\text{OPT}} \cap M_o} (L_i^*)^p \right)^{1/p} &- \left(\sum_{i \in M_{\text{OPT}} \cap M_o} (L_i^*)^p \right)^{1/p} \\ &\geq \left(\frac{\Delta_3\Phi}{10} + 2(\Delta_3\Phi) \right)^{1/p} - (2(\Delta_3\Phi))^{1/p} \geq \frac{(2(\Delta_3\Phi))^{1/p}}{40p}.\end{aligned}$$

The two last equations imply $\frac{(2(\Delta_3\Phi))^{1/p}}{40p} \leq \beta^{1/p}$, which implies $2(\Delta_3\Phi) = O(m \log m)$. Next, consider $2(\Delta_3\Phi) \leq \sum_{i \in M_{\text{OPT}} \cap M_o} L_i^{*p}$. Then, we have

$$2(\Delta_3\Phi) \leq \sum_{i \in M_{\text{OPT}} \cap M_o} L_i^{*p} \leq \Phi_0 + \Delta_o\Phi + \Delta_3\Phi,$$

which implies that $\Delta_3\Phi \leq \Phi_0 + \Delta_o\Phi = O(m \log m)$ by Lemmas 11, 13, 14, and 15. \square

The overall bound on the potential now follows from Lemmas 11, 13, 14, 15, and 16.

Theorem 17. *At the end of the algorithm, the potential is $O(m \log m) = O((40p)^p \beta)$.*

Bounding the cost and objective function. Having provided a bound on the potential function, we now relate it to the fractional cost and ℓ_p -norm of machine loads using Lemma 18 and Lemma 19 respectively.

Lemma 18. *For each partially open machine i , $\Delta y_{ij} p_{ij} \leq \Delta x_i c_i^{1/p}$.*

Proof. In each update step of a partially open machine i ,

$$\Delta y_{ij} p_{ij} = \frac{p_{ij} x_i}{\psi_{ij} N} = \frac{p_{ij} \Delta x_i c_i}{\psi_{ij}} \leq \frac{p_{ij} \Delta x_i c_i}{c_i^{(p-1)/p} p_{ij}} = \Delta x_i c_i^{1/p}. \quad \square$$

\square

Lemma 19. *For each partially open machine i , $\Delta y_{ij} p_{ij}^p \leq \Delta x_i c_i$.*

Opening Machines: For every machine i whose blue copy is closed, open it with probability (w/p) $\min\left(\frac{\alpha(x_i(j)-x_i(j-1))}{1-\alpha \cdot x_i(j-1)}, 1\right)$. (Eqn. 8 is satisfied by this rule using conditional probabilities.)

Assigning Job j :

- if $\sum_{i \in M^{(1)}(j)} y_{ij} \geq \frac{1}{2}$, then assign to blue copy of $i \in M^{(1)}(j)$ w/p $\frac{y_{ij}}{\sum_{i \in M^{(1)}(j)} y_{ij}}$,
- else if $\sum_{i \in M_o^{(0)}(j)} z_{ij} \geq 1$, then assign to blue copy of $i \in M_o^{(0)}(j)$ w/p $\frac{z_{ij}}{\sum_{i \in M_o^{(0)}(j)} z_{ij}}$,
- else assign to red copy of $i^* = \arg \min_{i \in M} ((\hat{L}_i + p_{ij})^p - \hat{L}_i^p)$ after opening it.

Algorithm 2: Assignment of a Single Job by the Integer Algorithm

Proof. In each update step of a partially open machine i ,

$$\Delta y_{ij} p_{ij}^p = \frac{p_{ij}^p x_i}{\psi_{ij} N} = \frac{p_{ij}^p \Delta x_i c_i}{\psi_{ij}} \leq \frac{p_{ij}^p \Delta x_i c_i}{p_{ij}^p} = \Delta x_i c_i. \quad \square$$

Finally, we give the overall bound for the fractional solution.

Theorem 20. *For the fractional solution, the objective (fractional ℓ_p -norm of loads) is bounded by $O((40p)^p \beta)$ and the total cost is bounded by $O(m \log m)$.*

Proof. The first term in the fractional objective is bounded using Lemma 18. If $x_i < 1$, then

$$\sum_j \left(\frac{y_{ij} p_{ij}}{x_i} \right)^p x_i \leq \left(\frac{c_i^{1/p} x_i}{x_i} \right)^p x_i = c_i x_i = \Phi_i.$$

On the other hand, if $x_i = 1$, then

$$\sum_j \left(\frac{y_{ij} p_{ij}}{x_i} \right)^p x_i \leq (c_i^{1/p} + \sum_{j \in J_1^{(i)}} y_{ij} p_{ij})^p = \tilde{L}_i^p \leq \Phi_i.$$

The second term in the fractional objective is bounded using Lemma 19:

$$\sum_j y_{ij} p_{ij}^p = \sum_{j \in J_0^{(i)}} y_{ij} p_{ij}^p + \sum_{j \in J_1^{(i)}} y_{ij} p_{ij}^p \leq c_i x_i + \sum_{j \in J_1^{(i)}} y_{ij} p_{ij}^p \leq \Phi_i.$$

Summing over all machines, the fractional objective is at most $2\Phi = O((40p)^p \beta)$. Since for all machines, $c_i x_i \leq \Phi_i$, the total cost is also bounded by the potential function, which is $O(m \log m)$ by Theorem 17. \square

4 Online Rounding for UMSC with ℓ_p norm

There are two decisions that an integer algorithm must make on receiving a new job j . First, it needs to decide the set of machines that it needs to open. Note that since decisions are irrevocable in the online model, the open machines form a monotonically growing set over time. Next, the algorithm must decide which among the open machines should it assign job j to. As we describe below, both these decisions are made by the integer algorithm based on the fractional solution that it maintains using the algorithm given in the previous section. Following nomenclature established by Alon *et al* [2], we call this process of producing an integer solution online based on a monotonically evolving fractional solution an *online randomized rounding* procedure.

To simplify the analysis later, we will consider two copies of each machine: a *blue* copy and a *red* copy. Note that this is without loss of generality, up to a constant factor loss in the competitive ratio for both the

cost and ℓ_p -norm objectives. First, we define a randomized process that controls the opening of blue copies of machines in the integer algorithm. Let $M_o(j)$ denote the set of machines whose blue copies are open after job j has been assigned, and $X_i(j)$ be an indicator random variable whose value is 1 if machine $i \in M_o(j)$ and 0 otherwise. Let $x_i(j)$ be the value of variable x_i in the fractional solution after job j has been completely assigned (fractionally). The integer algorithm maintains the invariant

$$\mathbb{P}[X_i(j) = 1] = \min(\alpha \cdot x_i(j), 1) \text{ for some parameter } \alpha \text{ that we will set later.} \quad (8)$$

using the rule given in Algorithm 2. Next, we need to assign job j to one of the open machines. We partition the set of machines M into two sets based on the fractional solution: $M^{(0)}(j)$ represents machines i such that $x_i(j) < \frac{1}{\alpha}$ and $M^{(1)}(j)$ represents machines i such that $x_i(j) \geq \frac{1}{\alpha}$. Note that after job j , the blue copies of all machines in $M^{(1)}(j)$ are open (by Eqn. 8). On the other hand, the blue copies of some subset of machines in $M^{(0)}(j)$ are open; call this subset $M_o^{(0)}(j)$, i.e. $M_o(j) = M_o^{(0)}(j) \cup M^{(1)}(j)$. In addition let, $z_{ij} = \frac{4y_{ij}}{\alpha \cdot x_i}$ and \hat{L}_i be the current sum of processing times of all jobs assigned to the red copy of machine i . The assignment rule for job j is given in Algorithm 2.

4.1 Analysis

First, we argue about the expected cost of the solution. To bound the cost of red copies, we show that Case 3 has low probability.

Lemma 21. *For any job j , the probability of case 3 is at most $\exp(-\alpha/48)$.*

Proof. Consider a machine $i \in M^{(0)}(j)$, i.e. $x_i(j) < \frac{1}{\alpha}$. Such a machine is open after job j with probability $\alpha x_i(j)$. Let us define a corresponding random variable

$$Z_{ij} = \begin{cases} z_{ij} & \text{if } i \in M_o^{(0)}(j) \\ 0 & \text{otherwise.} \end{cases}$$

We need to bound the probability that $\sum_{i \in M^{(0)}(j)} Z_{ij} < 1$.

First, we observe that $Z_{ij} \leq \frac{4y_{ij}}{\alpha \cdot x_i(j)} \leq \frac{8}{\alpha}$ since $y_{ij} \leq x_i(j)$. Now, consider random variable

$$\tilde{Z}_{ij} = \begin{cases} \frac{8}{\alpha} & \text{with probability } \frac{\alpha y_{ij}}{2} \\ 0 & \text{otherwise.} \end{cases}$$

Note that the expectations of Z_{ij} and \tilde{Z}_{ij} are identical and both have only one non-zero value in their support, but Z_{ij} has a strictly smaller range. Therefore, any tail bounds that apply to \tilde{Z}_{ij} also apply to Z_{ij} . Further, note that

$$\mathbb{E} \left[\sum_{i \in M^{(0)}(j)} Z_{ij} \right] = \mathbb{E} \left[\sum_{i \in M^{(0)}(j)} \tilde{Z}_{ij} \right] = 4 \sum_{i \in M^{(0)}(j)} y_{ij} \geq 2.$$

Therefore, by Chernoff-Hoeffding bounds (e.g., [29]),

$$\begin{aligned} \mathbb{P} \left[\sum_{i \in M^{(0)}(j)} Z_{ij} < 1 \right] &\leq \mathbb{P} \left[\sum_{i \in M^{(0)}(j)} \tilde{Z}_{ij} < 1 \right] = \mathbb{P} \left[\sum_{i \in M^{(0)}(j)} \frac{\alpha}{8} \tilde{Z}_{ij} < \frac{\alpha}{8} \right] \\ &\leq \exp \left(-\frac{\frac{1}{2^2} \cdot \frac{2 \cdot \alpha}{8}}{3} \right) = \exp(-\alpha/48). \end{aligned}$$

□

We choose $\alpha = 48 \ln(mn)$ to obtain the following corollary. (For now, $\alpha = 48 \ln n$ would have sufficed but we will need $\alpha \geq \ln m$ in a later step.)

Corollary 22. *For any job j , the probability of case 3 is at most $\frac{1}{mn}$.*

Recall that the cost of each individual machine is at most m . Using linearity of expectation and the above corollary, we can now claim that the expected cost of red copies of machines is at most m . Similarly, using linearity of expectation and Eqn. 8, we can claim that the expected cost of blue copies of machines is $\sum_{i \in M} c_i \alpha x_i \leq \alpha \Phi$. Overall, we get the following bound for the cost of machines opened by the integer algorithm.

Lemma 23. *The total expected cost of machines in the integer algorithm is at most $(\alpha + 1)\Phi = O(\ln(mn))\Phi$.*

We are now left to bound the expected ℓ_p -norm of machine loads. First, we obtain a bound for red copies of machines. Note that the assignment of jobs in Case 3 follows a greedy algorithm assuming all machines are open. Therefore, the analysis of the ℓ_p -norm of the red machines follows directly from the corresponding analysis without startup costs [17].

Lemma 24 ([17]). *The competitive ratio of the ℓ_p -norm of the red copies of machines is $O(p)$.*

Finally, we will bound the ℓ_p -norm of blue copies of machines. Let us define an indicator random variable

$$Y_{ij} = \begin{cases} 1 & \text{if job } j \text{ is assigned to the blue copy of machine } i \text{ in the integer solution} \\ 0 & \text{otherwise.} \end{cases}$$

Then, the ℓ_p^p -norm of the integer solution can be written as $\sum_{i \in M} (\sum_{j \in J} Y_{ij} p_{ij})^p$. We will bound the expected ℓ_p^p -norm of each machine individually, and then use linearity of expectation over all the machines.

Our main technical tool in bounding the expected ℓ_p^p -norm of a single machine will be the following theorem (see e.g., [24] for a proof).

Theorem 25. *Let W_1, W_2, \dots, W_n be independent non-negative random variables. Let $p > 1$ and $K_p = \Theta\left(\frac{p}{\log p}\right)$. Then,*

$$\mathbb{E} \left[\left(\sum_{j=1}^n W_j \right)^p \right] \leq (K_p)^p \cdot \max \left(\left(\sum_{j=1}^n \mathbb{E}[W_j] \right)^p, \sum_{j=1}^n \mathbb{E}[(W_j)^p] \right).$$

Ideally, we would like to use this theorem directly with $W_j = Y_{ij} p_{ij}$. This is indeed possible if $x_i(j) \geq \frac{1}{\alpha}$ since the assignment of such jobs j to machine i are independent of each other. However, the assignment of jobs j for which $x_i(j) < \frac{1}{\alpha}$ are *not* independent; they depend on each other via the random variable X_i which denotes whether machine i is open or not. Let j_i be the job that opened machine i , i.e. $X_i(j_i - 1) = 0$ and $X_i(j_i) = 1$. Conditioned on j_i , the variables Y_{ij} for $j \geq j_i$ are indeed independent. First, we will reduce the conditioning to a single indicator random variable. Define an indicator random variable $X_i = 1$ if and only if machine i is open in the integer solution after all n jobs have been assigned. By Eqn. 8, $X_i = 1$ with probability $\min(\alpha x_i, 1)$, where x_i is the fractional variable after all n jobs have been (fractionally) assigned. Now, define a binary random variable \tilde{Y}_{ij} with the following properties:

- if $X_i = 0$, then $\tilde{Y}_{ij} = 0$,
- else if job j is not assigned via case 2, then $\tilde{Y}_{ij} = Y_{ij}$,
- else, $\tilde{Y}_{ij} = 1$ with probability z_{ij} ; furthermore, in this case, using shared randomness, we ensure that $\tilde{Y}_{ij} = 1$ whenever $Y_{ij} = 1$.

The last condition can be met since in case 2, $\mathbb{P}[Y_{ij} = 1] = \frac{z_{ij}}{\sum_{i \in M_o^{(0)}(j)} z_{ij}} \leq z_{ij}$. Note that conditioned on $X_i = 1$, \tilde{Y}_{ij} for different jobs j are independent random variables. Furthermore, \tilde{Y}_{ij} stochastically dominates Y_{ij} , i.e. $Y_{ij} = 1$ implies $\tilde{Y}_{ij} = 1$. Therefore, it suffices to bound $(\sum_{j \in J} \mathbb{E}[\tilde{Y}_{ij} p_{ij}])^p$ and $\sum_{j \in J} \mathbb{E}[\tilde{Y}_{ij}] (p_{ij})^p$, conditioned on $X_i = 1$. In the next lemma, we bound the first term.

Lemma 26. *For any machine i , conditioned on the event $X_i = 1$, we have*

$$\left(\sum_{j \in J} \mathbb{E}[\tilde{Y}_{ij} p_{ij}] \right)^p \leq \left(5c_i^{1/p} + \sum_{j \in J_1^{(i)}} y_{ij} p_{ij} \right)^p.$$

Proof. We consider two phases for machine i : $x_i < 1$ and $x_i = 1$. Recall that the jobs assigned in the first phase are denoted $J_0^{(i)}$ and those in the second phase are denoted $J_1^{(i)}$. First, we note that for jobs $j \in J_1^{(i)}$, $\mathbb{E}[\tilde{Y}_{ij}] \leq y_{ij}$. Therefore, $\sum_{j \in J_0^{(i)}} \mathbb{E}[\tilde{Y}_{ij} p_{ij}] \leq \sum_{j \in J_0^{(i)}} y_{ij} p_{ij}$. On the other hand, for jobs $j \in J_0^{(i)}$, we need to distinguish between jobs assigned via case 2 while $x_i(j) < \frac{1}{\alpha}$ (call this set $J_0^{(i)}(2)$) and those that are assigned via case 3 after $x_i(j) \geq \frac{1}{\alpha}$ (call this set $J_0^{(i)}(3)$). Then,

$$\begin{aligned} \sum_{j \in J_0^{(i)}} \mathbb{E}[\tilde{Y}_{ij} p_{ij}] &\leq \sum_{j \in J_0^{(i)}(2)} z_{ij} p_{ij} + \sum_{j \in J_0^{(i)}(3)} y_{ij} p_{ij} \leq \frac{4}{\alpha} \sum_{j \in J_0^{(i)}(2)} \frac{y_{ij} p_{ij}}{x_i(j)} + \sum_{j \in J_0^{(i)}(3)} y_{ij} p_{ij} \\ &\leq \frac{4}{\alpha} \int_{1/m}^{1/\alpha} c_i^{1/p} \frac{dx}{x} + c_i^{1/p} \left(1 - \frac{1}{\alpha} \right) \leq \frac{4}{\alpha} \int_{1/m}^1 c_i^{1/p} \frac{dx}{x} + c_i^{1/p} \leq 5c_i^{1/p}, \end{aligned}$$

since $\alpha = 48 \ln(mn) \geq \ln m$. Combining all jobs,

$$\left(\sum_{j \in J} \mathbb{E}[\tilde{Y}_{ij} p_{ij}] \right)^p \leq \left(5c_i^{1/p} + \sum_{j \in J_1^{(i)}} y_{ij} p_{ij} \right)^p.$$

□

Next, we bound $\sum_{j \in J} \mathbb{E}[\tilde{Y}_{ij}] (p_{ij})^p$, conditioned on $X_i = 1$.

Lemma 27. *For any machine i , conditioned on the event $X_i = 1$, we have*

$$\sum_{j \in J} \mathbb{E}[\tilde{Y}_{ij}] (p_{ij})^p \leq 5c_i + \sum_{j \in J_1^{(i)}} y_{ij} (p_{ij})^p.$$

Proof. As in the previous proof, we consider two phases for machine i : $x_i < 1$ and $x_i = 1$. As earlier, the set of jobs assigned in the first phase is denoted $J_0^{(i)}$ and that assigned in the second phase is denoted $J_1^{(i)}$. First, we note that for jobs $j \in J_1^{(i)}$, $\mathbb{E}[\tilde{Y}_{ij}] \leq y_{ij}$. Therefore, $\sum_{j \in J_0^{(i)}} \mathbb{E}[\tilde{Y}_{ij}] (p_{ij})^p \leq \sum_{j \in J_0^{(i)}} y_{ij} (p_{ij})^p$. On the other hand, for jobs $j \in J_0^{(i)}$, we need to distinguish between jobs assigned via case 2 while $x_i(j) < \frac{1}{\alpha}$ (called $J_0^{(i)}(2)$) and those that are assigned via case 3 after $x_i(j) \geq \frac{1}{\alpha}$ (called $J_0^{(i)}(3)$). Then,

$$\begin{aligned} \sum_{j \in J_0^{(i)}} \mathbb{E}[\tilde{Y}_{ij}] (p_{ij})^p &\leq \sum_{j \in J_0^{(i)}(2)} z_{ij} (p_{ij})^p + \sum_{j \in J_0^{(i)}(3)} y_{ij} (p_{ij})^p \\ &\leq \frac{4}{\alpha} \sum_{j \in J_0^{(i)}(2)} \frac{y_{ij} (p_{ij})^p}{x_i(j)} + \sum_{j \in J_0^{(i)}(3)} y_{ij} (p_{ij})^p \leq \frac{4}{\alpha} \int_{1/m}^{1/\alpha} c_i \frac{dx}{x} + c_i (1 - 1/\alpha) \\ &\leq \frac{4}{\alpha} \int_{1/m}^1 c_i \frac{dx}{x} + c_i \leq 5c_i, \end{aligned}$$

since $\alpha = 48 \ln(mn) \geq \ln m$. Combining all jobs,

$$\sum_{j \in J} \mathbb{E}[\tilde{Y}_{ij}] (p_{ij})^p \leq 5c_i + \sum_{j \in J_1^{(i)}} y_{ij} (p_{ij})^p.$$

□

Finally, we apply Theorem 25 to Lemmas 26 and 27, and remove the conditioning on X_i .

Theorem 28. *For any machine i ,*

$$\mathbb{E} \left[\left(\sum_{j \in J} Y_{ij} p_{ij} \right)^p \right] \leq ((5\alpha)^{1/p} K_p)^p \Phi_i,$$

where $K_p = \theta \left(\frac{p}{\log p} \right)$.

Proof. For any machine i , conditioned on the event $X_i = 1$, we have

$$\begin{aligned} \mathbb{E} \left[\left(\sum_{j \in J} Y_{ij} p_{ij} \right)^p \right] &\leq \mathbb{E} \left[\left(\sum_{j \in J} \tilde{Y}_{ij} p_{ij} \right)^p \right] \quad (\text{since } \tilde{Y}_{ij} \text{ stochastically dominates } Y_{ij}) \\ &\leq (K_p)^p \max \left(\left(\sum_{j \in J} \mathbb{E}[\tilde{Y}_{ij} p_{ij}] \right)^p, \sum_{j \in J} \mathbb{E}[\tilde{Y}_{ij}] (p_{ij})^p \right) \quad (\text{using Theorem 25}) \\ &\leq (K_p)^p \max \left(\left(5c_i^{1/p} + \sum_{j \in J_1^{(i)}} y_{ij} p_{ij} \right)^p, 5c_i + \sum_{j \in J_1^{(i)}} y_{ij} (p_{ij})^p \right). \end{aligned}$$

We now have three cases. First, suppose machine i satisfies $x_i = 1$ after all the jobs have been fractionally assigned. Then $X_i = 1$ deterministically, and the above inequality holds unconditionally. Therefore,

$$\begin{aligned} \mathbb{E} \left[\left(\sum_{j \in J} Y_{ij} p_{ij} \right)^p \right] &\leq (K_p)^p \max \left(\left(5c_i^{1/p} + \sum_{j \in J_1^{(i)}} y_{ij} p_{ij} \right)^p, 5c_i + \sum_{j \in J_1^{(i)}} y_{ij} (p_{ij})^p \right) \\ &\leq (5K_p)^p \Phi_i. \end{aligned}$$

Next, consider machines i such that $\frac{1}{\alpha} \leq x_i < 1$ after all the jobs have been fractionally assigned. As in the previous case, $X_i = 1$ deterministically, and therefore the above inequality holds unconditionally. However, for such machines, $J_1^{(i)} = \emptyset$. Therefore,

$$\begin{aligned} \mathbb{E} \left[\left(\sum_{j \in J} Y_{ij} p_{ij} \right)^p \right] &\leq (K_p)^p \max \left((5c_i^{1/p})^p, 5c_i \right) \\ &= 5 (K_p)^p c_i \leq ((5\alpha)^{1/p} K_p)^p \Phi_i. \end{aligned}$$

Finally, consider machines i such that $x_i < \frac{1}{\alpha}$ after all the jobs have been fractionally assigned. As in the previous case, for such machines, $J_1^{(i)} = \emptyset$. However, $X_i = 1$ with probability αx_i . Therefore,

$$\begin{aligned} \mathbb{E} \left[\left(\sum_{j \in J} Y_{ij} p_{ij} \right)^p \right] &= \mathbb{E} \left[\left(\sum_{j \in J} Y_{ij} p_{ij} \right)^p \middle| X_i = 1 \right] \cdot \mathbb{P}[X_i = 1] \\ &\leq (K_p)^p (5c_i) \alpha x_i \leq ((5\alpha)^{1/p} K_p)^p \Phi_i. \end{aligned}$$

□

This completes the proof of Theorem 4.

Opening Machines: For every machine i whose blue copy is closed, open it with probability $\min\left(\frac{\alpha(x_i(j)-x_i(j-1))}{1-\alpha \cdot x_i(j-1)}, 1\right)$. (Eqn. 8 is satisfied by this rule using conditional probabilities.)

Assigning Job j :

- if $M_o(j) \cap M_{1/2}(j) \neq \emptyset$, then assign to blue copy of any machine in $M_o(j) \cap M_{1/2}(j)$,
- else assign to red copy of machine $i^* = \arg \min_{i \in M} p_{ij}$, after opening it if necessary.

Algorithm 3: Assignment of a Single Job by the Integer Algorithm for the ℓ_1 -norm

5 Online Rounding for UMSC with ℓ_1 norm

We now present an online rounding algorithm specifically tailored to the important special case of $p = 1$, i.e., the ℓ_1 -norm. The rule for opening machines is identical (with a smaller value of α that we will shortly calculate) to the rounding algorithm for general p . However, the assignment rule for a job is now simpler and is given in Algorithm 3. Here, $M(j)$ denotes the machines sorted in non-decreasing order of p_{ij} and $M_{1/2}(j)$ is the minimal prefix of $M(j)$ that satisfies $\sum_{i \in M_{1/2}(j)} y_{ij} \geq 1/2$. As earlier, for clarity, we use two copies of each machine, a blue copy and a red copy, and let $M_o(j)$ be the machines whose blue copies are open after job j .

5.1 Analysis

First, we argue about the expected cost of the solution. To bound the cost of red copies, we show that Case 2 has low probability.

Lemma 29. *For any job j , the probability of case 2 is at most $\exp(-\alpha/4)$.*

Proof. Note that

$$\sum_{i \in M_{1/2}(j)} x_i(j) \geq \sum_{i \in M_{1/2}(j)} \frac{y_{ij}}{2} \geq \frac{1}{4}.$$

Therefore, the probability of case 2 is

$$\begin{aligned} \prod_{i \in M_{1/2}(j)} (1 - \alpha x_i(j)) &\leq \left(1 - \frac{\alpha \sum_{i \in M_{1/2}(j)} x_i(j)}{k}\right)^k \\ &\leq \exp\left(-\alpha \sum_{i \in M_{1/2}(j)} x_i(j)\right) \leq \exp(-\alpha/4). \end{aligned}$$

□

We choose $\alpha = 4 \ln n$ to obtain the following corollary.

Corollary 30. *For any job j , the probability of case 2 is at most $\frac{1}{n}$.*

Recall that the cost of each individual machine is at most m . Using linearity of expectation and the above corollary, we can now claim that the expected cost of red copies of machines is at most m . Similarly, using linearity of expectation and Eqn. 8, we can claim that the expected cost of blue copies of machines is $\sum_{i \in M} c_i \alpha x_i \leq \alpha \Phi$. Overall, we get the following bound for the cost of machines opened by the integer algorithm.

Lemma 31. *The total expected cost of machines in the integer algorithm is at most $(\alpha + 1)\Phi = O(\ln n)\Phi$.*

We are now left to bound the ℓ_1 -norm of the assignment. First, consider the red copies of machines. Note that the assignment of jobs in Case 2 follows a greedy algorithm assuming all machines are open. Therefore, the ℓ_1 -norm of red copies of machines is optimal. The next lemma complements this observation by bounding the ℓ_1 -norm of blue copies of machines.

Lemma 32. *The expected ℓ_1 -norm of blue copies of machines is at most 2Φ .*

Proof. Suppose we assigned job j to the blue copy of machine \hat{i} . Also, let $k(j)$ be the last machine in the prefix $M_{1/2}(j)$ and let $\overline{M}_{1/2}(j) = (M \setminus M_{1/2}(j)) \cup \{k(j)\}$. Then, we have $\sum_{i \in \overline{M}_{1/2}(j)} y_{ij} \geq 1/2$ by minimality of the prefix $M_{1/2}(j)$ and $p_{\hat{i}j} \leq p_{ij}$ for all machines $i \in \overline{M}_{1/2}(j)$. Then, the increase in ℓ_1 -norm of the integer solution is $p_{\hat{i}j}$ whereas the corresponding increase in Φ for the fractional solution is

$$\sum_{i \in M} y_{ij} p_{ij} \geq \sum_{i \in \overline{M}_{1/2}(j)} y_{ij} p_{ij} \geq p_{\hat{i}j} \sum_{i \in \overline{M}_{1/2}(j)} y_{ij} \geq \frac{p_{\hat{i}j}}{2}.$$

The lemma now follows by summing over all jobs. □

This completes the proof of Theorem 6.

References

- [1] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. A general approach to online network optimization problems. *ACM Transactions on Algorithms*, 2(4):640–660, 2006.
- [2] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. The online set cover problem. *SIAM J. Comput.*, 39(2):361–370, 2009.
- [3] Antonios Antoniadis, Sungjin Im, Ravishankar Krishnaswamy, Benjamin Moseley, Viswanath Nagarajan, Kirk Pruhs, and Cliff Stein. Hallucination helps: Energy efficient virtual circuit routing. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1141–1153, 2014.
- [4] James Aspnes, Yossi Azar, Amos Fiat, Serge A. Plotkin, and Orli Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *J. ACM*, 44(3):486–504, 1997.
- [5] Baruch Awerbuch, Yossi Azar, Edward F. Grove, Ming-Yang Kao, P. Krishnan, and Jeffrey Scott Vitter. Load balancing in the ℓ_p norm. In *FOCS*, pages 383–391, 1995.
- [6] Yossi Azar. On-line load balancing. In *Online Algorithms*, pages 178–195, 1996.
- [7] Yossi Azar, Umang Bhaskar, Lisa K. Fleischer, and Debmalya Panigrahi. Online mixed packing and covering. In *SODA*, 2013.
- [8] Yossi Azar and Amir Epstein. Convex programming for scheduling unrelated parallel machines. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 331–337, 2005.
- [9] Yossi Azar, Joseph Naor, and Raphael Rom. The competitiveness of on-line assignments. *J. Algorithms*, 18(2):221–237, 1995.
- [10] Nikhil Bansal, Niv Buchbinder, Aleksander Madry, and Joseph Naor. A polylogarithmic-competitive algorithm for the k -server problem. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 267–276, 2011.

- [11] Nikhil Bansal, Niv Buchbinder, and Joseph Naor. Towards the randomized k-server conjecture: A primal-dual approach. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 40–55, 2010.
- [12] Nikhil Bansal, Niv Buchbinder, and Joseph Naor. A primal-dual randomized algorithm for weighted paging. *J. ACM*, 59(4):19, 2012.
- [13] Nikhil Bansal, Niv Buchbinder, and Joseph Naor. Randomized competitive algorithms for generalized caching. *SIAM J. Comput.*, 41(2):391–414, 2012.
- [14] Nikhil Bansal, Anupam Gupta, Ravishankar Krishnaswamy, Viswanath Nagarajan, Kirk Pruhs, and Cliff Stein. Multicast routing for energy minimization using speed scaling. In *Design and Analysis of Algorithms - First Mediterranean Conference on Algorithms, MedAlg 2012, Kibbutz Ein Gedi, Israel, December 3-5, 2012. Proceedings*, pages 37–51, 2012.
- [15] Ken Birman, Gregory Chockler, and Robbert van Renesse. Toward a cloud computing research agenda. *SIGACT News*, 40(2):68–80, 2009.
- [16] Niv Buchbinder and Joseph Naor. Online primal-dual algorithms for covering and packing. *Math. Oper. Res.*, 34(2):270–286, 2009.
- [17] Ioannis Caragiannis. Better bounds for online load balancing on unrelated machines. In *SODA*, pages 972–981, 2008.
- [18] Ioannis Caragiannis, Michele Flammini, Christos Kaklamanis, Panagiotis Kanellopoulos, and Luca Moscardelli. Tight bounds for selfish and greedy load balancing. In *ICALP (I)*, pages 311–322, 2006.
- [19] Lisa Fleischer. Data center scheduling, generalized flows, and submodularity. In *ANALCO*, pages 56–65, 2010.
- [20] Naveen Garg and Jochen Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *39th Annual Symposium on Foundations of Computer Science, FOCS '98, November 8-11, 1998, Palo Alto, California, USA*, pages 300–309, 1998.
- [21] Anupam Gupta, Ravishankar Krishnaswamy, and Kirk Pruhs. Online primal-dual for non-linear optimization with applications to speed scaling. In *Approximation and Online Algorithms - 10th International Workshop, WAOA 2012, Ljubljana, Slovenia, September 13-14, 2012, Revised Selected Papers*, pages 173–186, 2012.
- [22] Mohammad Taghi Hajiaghayi, Vahid Liaghat, and Debmalya Panigrahi. Online node-weighted steiner forest and extensions via disk paintings. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 558–567, 2013.
- [23] MohammadTaghi Hajiaghayi, Vahid Liaghat, and Debmalya Panigrahi. Near-optimal online algorithms for prize-collecting steiner problems. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 576–587, 2014.
- [24] W.B. Jhonson, G. Scheechtman, and J. Zinn. Best constants in moment inequalities for linear combinations of independent and exchangeable random variables. *Ann. Probab.*, 1:234–253, 1985.
- [25] Samir Khuller, Jian Li, and Barna Saha. Energy efficient scheduling via partial shutdown. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1360–1372, 2010.

- [26] Simon Korman. On the use of randomization in the online set cover problem. *M.S. thesis, Weizmann Institute of Science*, 2005.
- [27] V. S. Anil Kumar, Madhav V. Marathe, Srinivasan Parthasarathy, and Aravind Srinivasan. Approximation algorithms for scheduling on multiple machines. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*, pages 254–263, 2005.
- [28] Jian Li and Samir Khuller. Generalized machine activation problems. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 80–94, 2011.
- [29] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1997.
- [30] Joseph Naor, Debmalya Panigrahi, and Mohit Singh. Online node-weighted steiner tree and related problems. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 210–219, 2011.
- [31] Serge A. Plotkin, David B. Shmoys, and va Tardos. Fast approximation algorithms for fractional packing and covering problems, 1995.
- [32] Kirk Pruhs, Jiri Sgall, and Eric Torng. Online scheduling. *Handbook of scheduling: algorithms, models, and performance analysis*, pages 15–1, 2004.
- [33] Jiri Sgall. On-line scheduling. In *Online Algorithms*, pages 196–231, 1996.
- [34] Jiri Sgall. Online scheduling. In *Algorithms for Optimization with Incomplete Information, 16.-21. January 2005*, 2005.
- [35] Wikipedia. Hölder’s inequality — wikipedia, the free encyclopedia, 2013.

Appendix

A Lower bound for OMPC with the ℓ_p norm objective

We adapt the example in Azar *et al.* [7] for the ℓ_∞ norm and analyze it for the ℓ_p norm. For parameters p and d , the example uses $r(\geq 2^p)$ packing constraints, each with at most $\hat{d} = d \log r$ variables and at most $2d$ (which is $< \hat{d}$) variables in any covering constraint. The example uses $2(r-1)$ pairwise disjoint sets (blocks) of d variables. We use B_i to refer to the i th block. In [7] there is a procedure of revealing covering constraints to two blocks such that at least one block has a weight of at least $H_d/2$, where H_d refers to the d th harmonic number, and there is feasible solution with total weight of 1 to one of the blocks.

The packing constraints are represented as follows: a complete binary tree with r leaf nodes. Each node in this tree except the root corresponds to a block, and no two nodes correspond to the same block. Our packing constraints correspond to the leaf nodes, with packing constraint k being $\sum(\cup_{i \in Q_k} B_i) \leq \lambda$ where Q_k is the set of blocks encountered on the path from the root to the leaf node corresponding to packing constraint k . In the example, initially apply the procedure to the two blocks which are the children of the roots. Then, apply the procedure to the children of the block with the larger weight ($\geq H_d/2$) and so on, until it reaches to one of the leafs. It is easy to verify that for each $1 \leq i \leq \log r$ there exists $2^{\log r - 1 - i}$ packing constraints with $\lambda \geq i \cdot H_d/2$. In addition, there exists a feasible solution with $\lambda_k = 1$ for any k . This yields

a competitive ratio of at least (for $p \leq \log r$)

$$\left(\frac{(H_d/2 \cdot \log r)^p + \sum_{i=1}^{\log r} (H_d/2 \cdot i)^p 2^{\log r - 1 - i}}{r} \right)^{1/p} \approx H_d/2 \cdot \frac{p \log e}{e} = \theta(p \log(\hat{d}/\log r)).$$